International Journal of Technology (2015) 5: 864-871
ISSN 2086-9614

SIMULATING PORT EXPANSION PLANS USING AGENT BASED MODELLING

Ramadhianto Agung Wibowo[1], Akhmad Hidayatno[1], Komarudin[1], Armand Omar Moeis[1*]

[1] *System Engineering, Modelling and Simulation Laboratory, Department of Industrial Engineering, Faculty of Engineering, Universitas Indonesia, Kampus Baru UI Depok, Depok 16424, Indonesia*

(Received: June 2015 / Revised: September 2015 / Accepted: October 2015)

ABSTRACT

Jakarta International Container Terminal (JICT), which is the main international port of Indonesia, is overcrowded and is going to be expanded. Various expansion plans are considered, so the aim of the current paper is to simulate the plans using an agent-based modeling approach. The plans are treated as scenarios in the agent-based simulation model. Then, the throughputs from all scenarios are compared. The comparison shows that some of the plans can support the JICT vision. The results can be used as an input for the decision maker to either choose one of the plans or renegotiate the plans to achieve results that greatly support the JICT vision.

Keywords: Agent based modelling; Logistic system; Port management; Simulation

1. INTRODUCTION

As the main gateway for container transportation to and from Indonesia, Jakarta International Container Terminal (JICT) has to handle the movement of millions of TEUs (twenty foot equivalent units) of containers every year. The current export capacity of JICT is 2.2 million TEUs per year, but JICT is barely able to handle the demand of the container movements, which reaches 2 million TEUs per year. It is predicted that this number is going to be doubled to 4 million TEUs per year by 2017. Therefore, the expansion of JICT is necessary. A number of plans has been submitted, and JICT must choose (or develop) an expansion plan. To help JICT in choosing the best plan, the aim of this paper is to develop an agent-based logistic model for each expansion plan. This will act as an input for the decision maker to either choose one of the plans or renegotiate the plans to achieve results that greatly support JICT's objectives.

The port is a complex system, as it has a high number of actors, each with its own behaviors. The actors interact with each other and this increases the port's complexity. The interactions among the actors shape the general behaviors of the port. Due to a high number of repetitive operations, a single change to a parameter value, which causes a small change on an individual basis, might result in a massive change to the port system.

To model such a complex system, an agent-based model (ABM) is chosen. The decision is made based on the similarities between the structure of the port system and the ABM. An ABM enables the design of actors resembling the real system without imposing unnecessary limitations on the actors' interactions.

* Corresponding author's email: armand@ie.ui.ac.id, Tel. +62-21-78888805, Fax. +62-21-78885656
Permalink/DOI: http://dx.doi.org/10.14716/ijtech.v6i5.1900

2. LITERATURE REVIEW

The use of a simulation in the design stage is not a new concept (Shephard et al., 2004). By using a simulation in the design stage, the system designer could experiment with various combinations of designs, which enables him or her to modify the design plan so that he or she can produce desirable outputs. Rather than experimenting with the actual system, using a simulation can enable a prediction of the output in a relatively short period, at a lower cost, and with fewer risks involved.

Bichou and Grey (2004) explain port as a complex system from a logistic viewpoint. Simulation approach allows for a better understanding of the behaviors of a complex system. Moreover, it can show the effects of the parameter value changes to the agent/individual behavior and this is also applies to a logistic system of a port (Van Gils et al., 2009).

To model such a complex system, an ABM approach is chosen based on its similarity to the complex port structure. An ABM excels in generating system-level behaviors based on the individual level (Knaak et al., 2006). Furthermore, ABM allows to easily incorporating the interactions in between agents in a model. This is in contrast with the flow-based simulation approach. This allows for more direct interactions and avoids unnecessary limitations in the agents' interactions.

Zouhaier et al. (2013) use an ABM approach to evaluate the performances of a hub port through several scenarios. They vary several parameters, i.e., the speed and the service time of the loading operation, to minimize the container processing time. In contrast, the current paper aims to evaluate several expansion plans using an ABM approach. The current paper focuses on evaluating the resultant capacity of those expansion plans to satisfy future container movement demands

3. METHODOLOGY

As a first step, we will design the simulation structure. Three plans need to be tested, and each plan will be translated as a change to the parameter needed to run the system. The input of the simulation system itself is forecasted demands. These demands will be simulated by a port simulation, which will output various measurement and behavior changes. The port simulation is controlled by various parameters, which will be modified for each run if the plan requires them to be changed. The outputs will then be compared with each other and will be presented to the stakeholder with accompanying data, such as the total investment required, predicted project span, etc.

The simulation is built using a programming platform based on the object oriented programming concept for ease of verification. The structure, which is similar to the individual concept of the real system, helps to compare the simulation processes and the real system. Among the available platforms, Java was chosen for its large community, which helped in finding solutions to various problems during the process of building this simulation. The software and libraries used are from Anylogic, which is based on Java. As it has limited support of Java methods, several required methods and objects of Java, which were not included in the Anylogic libraries, are invoked using the additional code space provided by Anylogic.

When building the port simulation, the real port system is inspected and the various working processes of individual actors are documented as a flowchart. This process did not require much effort, as each flowchart will be used as-is as a state chart of the agents.

The model is built on two-dimensional spaces as its environment. Technically, these spaces are other objects in the simulation programming. To prevent a misunderstanding, superclass and subclass will be used as references to the technical hierarchy, while above and below will be used as references to the conceptual hierarchy.

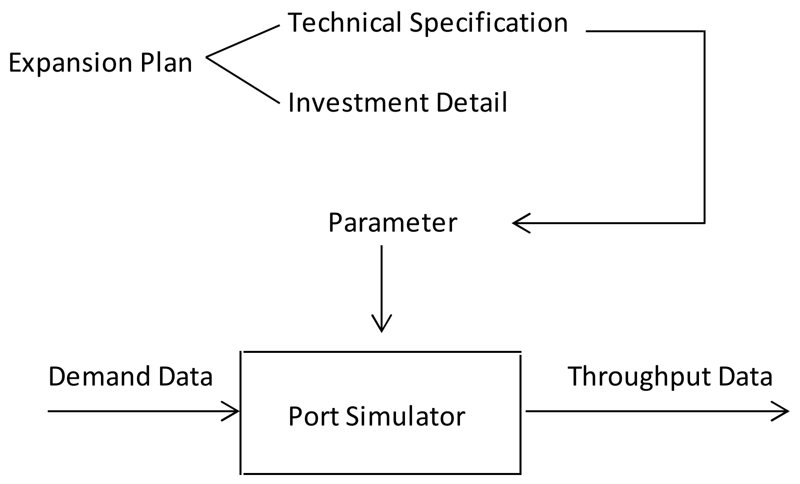

Figure 1 Input-process-output diagram of the port proposal simulation system

The *Main* object, which contains the space or environment for the agents, does not have a flow process chart. There are, however, various functions required as interactions between agents and the environment (world). These functions are built based on various interactions and processes between actors and the world outside the port system. Among them are the physical movement, time controller, object add-subtract mechanism, and pooling system for accommodating communications among agents not yet connected.

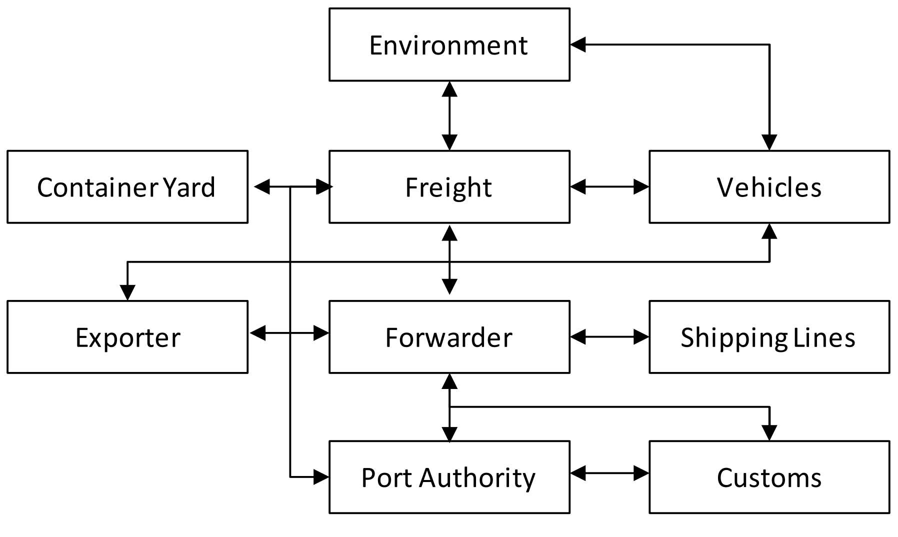

Figure 2 Interactions among agents/actors in a port logistic simulator

From the real system, actors are inspected and recreated in the simulator as agents. Their logistic processes are translated into a flowchart and used as their respective agents' state charts. The interactions among agents are mapped and categorized as limited and free interactions. An example of each, respectively, includes the interactions between the freight and its handler and the interactions among handlers. The parameters for the run/trigger of each state on the state chart are based on the real system flowchart. Two are time-based (process delay) and trigger-based.

To interact among each other, agents use communication methods based on messages, which are strings of data being passed among agents, and they consist of various objects, such as references to certain agents, strings, information, etc. To send a message, there are two requirements to invoke: the messages and the recipients. If necessary, it is possible to send a message to all individual agents of the same or different types. To do so, a superclass reference of the individual agents must be provided in the *Main* object. However, to send a message to a specific agent, the sender must have a static reference to that agent.

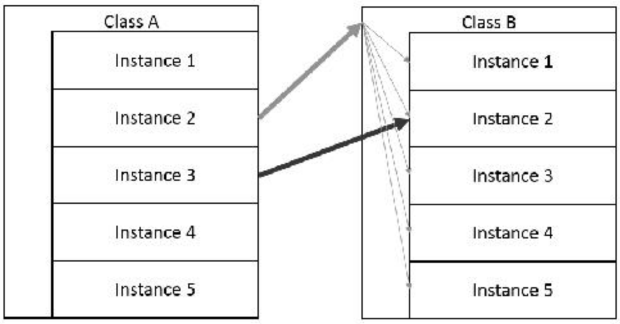

Figure 3 Methods of communication among agents: direct communication (red) and broadcast message (blue)

Agents can be classified into two types based on their cycle: the idle type and the exit type.

1. <u>Idle type</u>
An agent is classified as an idle type if it is still in the system after it completes its processes. The agent's number in the system is not affected by an everyday operation and will only be changed if the scenario requires it to do so.

2. <u>Exit type</u>
An agent is classified as an exit type if it exits the system after it completes its processes. The number of agents in the system varies every time, as decided by adding and subtracting such agents to and from the system. As they exit the system, the garbage collection mechanism will delete the instance from the heap or memory of the Java virtual memory. To prevent a lost count of the instances completed, just before they exit the system, a counting variable records the number of all exit-type agents in the *Main* object.

Afterwards, we will build the technical hierarchy of the agents. Using a class diagram, the structure of this diagram is built not based on each actor's role in the system, but rather on their technical parameters. The agents are classified into logistic actors, vehicles, and port facilities. Whenever a class becomes a subclass of another class, the subclass inherits all variables, methods, and parameters of its superclass.

After the process of each individual agent is described in and based on the state chart, the interactions among them are integrated as trigger mechanisms between state charts. By sending a message to a state chart, a transition could be triggered. Rather than explicitly stating the interactions among agents in the main object, the integration to the state chart trigger mechanism will avoid the unnecessary limitation of the interaction within the port simulator.

To facilitate communication between not-yet-connected agents, the main object is built with a pooling mechanism. It works as follows:

1) The first agent will add a static reference of itself to a list in the main object.

2) The first agent will then send a message to all instances of the second agent. Triggered by the message, all instances of the second agent check the list of all agents in the *Main* object.

If the second agent is found, the top agent reference will be retrieved. If not, they will return to the idle state.

1) The second agent will put the link into a local variable.
2) The second agent will send a message containing a static reference to itself to the first agent instance using the static reference to the first agent instance.
3) The first agent will store the link to the second agent instance to a local variable. The links are therein established.

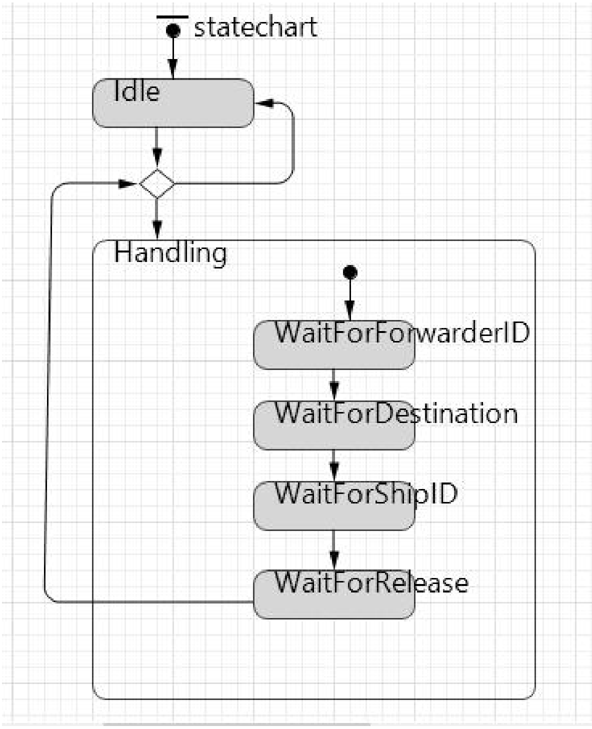

Figure 4 State chart example of agents who will idle after the process cycle is finished

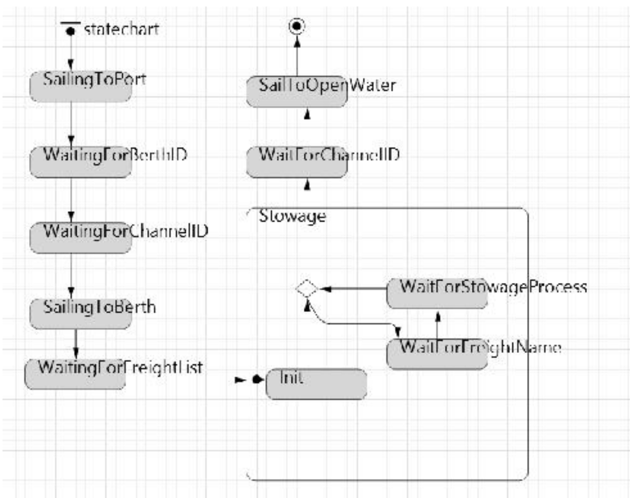

Figure 5 State chart example of agents who will exit the system after the process cycle is finished

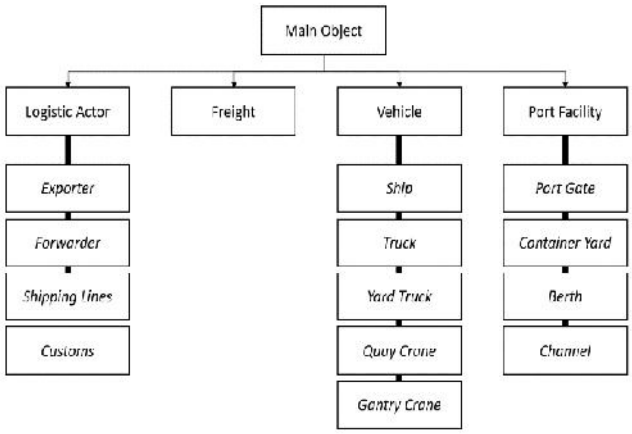

Figure 6 Class diagram of agents in a port logistics simulator

After the process is completed, a release order will be issued, which will set the local variable to "null." The links are then severed.

Upon assessment, no plan requires a fundamental process change between plans, so the port simulator itself will be the same for all post-implementation runs. There is, however, a fundamental change between the current condition and the post-implementation run. All plans involved use a dedicated channel for ingoing-outgoing traffic. The current condition, however, uses only a single channel for both ingoing-outgoing traffic, which requires a ship to take a turn and wait until the channel is completely cleared before going the other way.

Such a change involves modifying the state chart for the ship agent. The pooling system, which previously involved waiting for the whole channel to be cleared before the ship goes either way, is now changed to a pooling system for two different agents, the Entrance Channel and the Exit Channel. To use these channels, a ship no longer has to wait for the channel to be completely cleared, but rather, a distance-based hold is used. This allows the ship waiting time for the channel to be significantly reduced.

The simulation was then run using parameters based on the current system or derived from the expansion plan. Each run started on 1 January 2014 and will go to 31 December 2030. After each run, the throughputs of the system, which are measured as TEUs per year, are copied into the comparison table from the counting variable of the freight agent in the main object.

4. RESULTS AND DISCUSSION

By comparing investment plans, we obtained results as shown in Table 1.

Throughput-wise, there is no significant difference between each expansion plan. There is, however, a notable drop in the demand fulfillment level for all plans after 2020. Comparing each plan throughput rate increase for 2020 and beyond, it becomes evident that the new throughput threshold has been reached. From this point onward, the projected capacity of all alternatives will not be able to fulfill the forecasted demand, as depicted in Figure 7. Comparing solely each plan's throughput with each investment, the appealing plans for investors are (from the most appealing) the first plan, followed by the third and second plans.

Table 1 Projected throughput (in million TEUs per year) and demand fulfillment of each plan and its investment (in US$ Million)

Investment		0	US$ 617	US$ 1000	US$ 789	Fulfillment			
Year	FD	B	1	2	3	B	1	2	3
2014	3.355	2.159	3.308	3.319	3.316	64.35%	98.61%	98.92%	99.64%
2015	3.685	2.177	3.667	3.670	3.676	59.07%	99.50%	99.58%	99.65%
2016	3.850	2.189	3.869	3.825	3.849	56.86%	100.50%	99.35%	99.56%
2017	4.400	2.196	4.361	4.394	4.355	49.91%	99.11%	99.86%	99.91%
2018	4.510	2.204	4.511	4.499	4.510	48.86%	100.02%	99.75%	99.83%
2019	4.950	2.242	4.804	4.795	4.822	45.29%	97.06%	96.87%	98.11%
2020	5.225	2.249	4.848	4.857	4.862	43.04%	91.78%	92.95%	94.14%
2021	5.775	2.249	4.895	4.907	4.952	38.95%	84.77%	84.97%	86.06%
2022	6.050	2.280	4.927	4.933	4.940	37.68%	81.44%	81.53%	82.58%
2023	6.463	2.279	4.947	4.955	4.962	35.26%	76.55%	76.67%	77.65%
2024	6.655	2.293	4.969	5.027	4.986	34.46%	74.66%	75.53%	76.50%
2025	7.150	2.298	4.979	4.942	4.993	32.14%	69.64%	69.12%	70.01%
2026	7.700	2.300	5.015	5.018	5.026	29.87%	65.12%	65.17%	66.01%
2027	8.250	2.320	5.029	5.042	5.051	28.12%	60.95%	61.11%	61.90%
2028	8.800	2.330	5.054	5.060	5.065	26.48%	57.43%	57.50%	58.24%
2029	9.350	2.350	5.076	5.077	5.080	25.13%	54.28%	54.29%	54.99%
2030	9.900	2.366	5.092	5.096	5.122	23.90%	51.44%	51.47%	52.13%

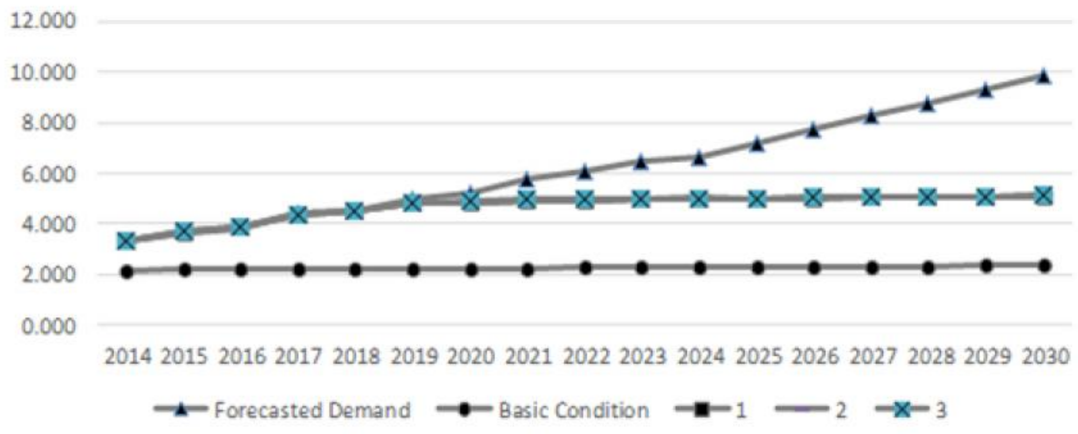

Figure 7 Capacity and demand over time in thousand TEUs per year

Comparing the effects of parameter changes between plans at the individual level on the system-level performance, it could be seen that such a change affects the system level, although the interactions among individuals are not defined at the system level. Observing this behavior is possible with agent-based technology, which enables us to trace the condition/state of each instance of all agents over time. Moreover, by observing the instances of each agent, an agent that causes a system bottleneck can be identified.

5. CONCLUSION

By comparing the throughput and the investment value, the most appealing expansion plan is the first plan, followed by the third and the second plans. However, it must be noted that the demand fulfillment rate will drop significantly after 2020. As the actual purpose of this expansion plan is to fulfill the container movement demands of JICT, an additional plan is required.

In the future, we would like to minimize the difference between the forecasted demands and the projected throughput. This difference, as the area between the forecasted demand line and the plan line, should become sales. This solution is likely to cause a major traffic jam in the port logistic system. Possible alternatives include, but are not limited to, a redesign of the proposal plan (following the initial concept of the simulation-based design), a formulation of a new expansion plan for effective implementation in 2020, and a minimization of the demand after 2020 by spreading it to other ports in the logistic system.

We also propose a further research direction regarding the effect of interactions among individual actors on port performance. As we are dealing with a logistic system, the discussion should not only be limited to the seaport, but should also include the airport and the dry port. Further research can be performed within the same logistic system, i.e., JICT.

6. REFERENCES

Bichou, K., Grey, R., 2004. A Logistic and Supply Management Approach to Port Performance Management. *Maritime Policy Management*, Volume 31, pp. 47–67

Knaak, N., Kruse, S., Page, B., 2006. An Agent-based Simulation Tools for Modelling Sustainable Logistic System. In: *Proceedings of the 7th International Congress on Environmental Modelling and Software (iEMSs), June 15-19, 2014, San Diego, California, USA*

Shephard, M.S., Beall, M.W., O'Bara, R.M., Webster, B.E., 2004. Toward Simulation-based Design. *Finite Elements in Analysis and Design*, Volume 10, pp. 1575–1598

Van Gils, M., Gerrits, L., Teisman, G.R., 2009. *Non Linear Dynamics in Port System*. A Complexity Theory for Public Policy. UK: Routledge

Zouhaier, H., Kebair, F., Serin, F., Said, L.B., 2013. Multi-agent Model and Simulation of a Hub Port Logistics. *Control, Decision, and Information Technologies*, pp. 671–676